10.3-datamanipulation

December 19, 2024

Data Manipulation and Analysis with Pandas Data manipulation and analysis are key tasks in any data science or data analysis project. Pandas provides a wide range of functions for data manipulation and analysis, making it easier to clean, transform, and extract insights from data. In this lesson, we will cover various data manipulation and analysis techniques using Pandas.

[1]:
```
import pandas as pd
```

[2]:
```
df=pd.read_csv('data.csv')
## fecth the first 5 rows
df.head(5)
```

[2]:

	Date	Category	Value	Product	Sales	Region
0	2023-01-01	A	28.0	Product1	754.0	East
1	2023-01-02	B	39.0	Product3	110.0	North
2	2023-01-03	C	32.0	Product2	398.0	East
3	2023-01-04	B	8.0	Product1	522.0	East
4	2023-01-05	B	26.0	Product3	869.0	North

[3]:
```
df.tail(5)
```

[3]:

	Date	Category	Value	Product	Sales	Region
45	2023-02-15	B	99.0	Product2	599.0	West
46	2023-02-16	B	6.0	Product1	938.0	South
47	2023-02-17	B	69.0	Product3	143.0	West
48	2023-02-18	C	65.0	Product3	182.0	North
49	2023-02-19	C	11.0	Product3	708.0	North

[4]:
```
df.describe()
```

[4]:

	Value	Sales
count	47.000000	46.000000
mean	51.744681	557.130435
std	29.050532	274.598584
min	2.000000	108.000000
25%	27.500000	339.000000
50%	54.000000	591.500000
75%	70.000000	767.500000
max	99.000000	992.000000

[5]:
```
df.dtypes
```

[5]:
```
Date          object
Category      object
Value        float64
Product       object
Sales        float64
Region        object
dtype: object
```

[6]:
```
## Handling Missing Values
df.isnull().any()
```

[6]:
```
Date        False
Category    False
Value        True
Product     False
Sales        True
Region      False
dtype: bool
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Date        0
Category    0
Value       3
Product     0
Sales       4
Region      0
dtype: int64
```

[9]:
```
df_filled=df.fillna(0)
```

[11]:
```
### filling missing values with the mean of the column
df['Sales_fillNA']=df['Sales'].fillna(df['Sales'].mean())
df
```

[11]:

	Date	Category	Value	Product	Sales	Region	Sales_fillNA
0	2023-01-01	A	28.0	Product1	754.0	East	754.000000
1	2023-01-02	B	39.0	Product3	110.0	North	110.000000
2	2023-01-03	C	32.0	Product2	398.0	East	398.000000
3	2023-01-04	B	8.0	Product1	522.0	East	522.000000
4	2023-01-05	B	26.0	Product3	869.0	North	869.000000
5	2023-01-06	B	54.0	Product3	192.0	West	192.000000
6	2023-01-07	A	16.0	Product1	936.0	East	936.000000
7	2023-01-08	C	89.0	Product1	488.0	West	488.000000
8	2023-01-09	C	37.0	Product3	772.0	West	772.000000
9	2023-01-10	A	22.0	Product2	834.0	West	834.000000

10	2023-01-11	B	7.0	Product1	842.0	North	842.000000
11	2023-01-12	B	60.0	Product2	NaN	West	557.130435
12	2023-01-13	A	70.0	Product3	628.0	South	628.000000
13	2023-01-14	A	69.0	Product1	423.0	East	423.000000
14	2023-01-15	A	47.0	Product2	893.0	West	893.000000
15	2023-01-16	C	NaN	Product1	895.0	North	895.000000
16	2023-01-17	C	93.0	Product2	511.0	South	511.000000
17	2023-01-18	C	NaN	Product1	108.0	West	108.000000
18	2023-01-19	A	31.0	Product2	578.0	West	578.000000
19	2023-01-20	A	59.0	Product1	736.0	East	736.000000
20	2023-01-21	C	82.0	Product3	606.0	South	606.000000
21	2023-01-22	C	37.0	Product2	992.0	South	992.000000
22	2023-01-23	B	62.0	Product3	942.0	North	942.000000
23	2023-01-24	C	92.0	Product2	342.0	West	342.000000
24	2023-01-25	A	24.0	Product2	458.0	East	458.000000
25	2023-01-26	C	95.0	Product1	584.0	West	584.000000
26	2023-01-27	C	71.0	Product2	619.0	North	619.000000
27	2023-01-28	C	56.0	Product2	224.0	North	224.000000
28	2023-01-29	B	NaN	Product3	617.0	North	617.000000
29	2023-01-30	C	51.0	Product2	737.0	South	737.000000
30	2023-01-31	B	50.0	Product3	735.0	West	735.000000
31	2023-02-01	A	17.0	Product2	189.0	West	189.000000
32	2023-02-02	B	63.0	Product3	338.0	South	338.000000
33	2023-02-03	C	27.0	Product3	NaN	East	557.130435
34	2023-02-04	C	70.0	Product3	669.0	West	669.000000
35	2023-02-05	B	60.0	Product2	NaN	West	557.130435
36	2023-02-06	C	36.0	Product3	177.0	East	177.000000
37	2023-02-07	C	2.0	Product1	NaN	North	557.130435
38	2023-02-08	C	94.0	Product1	408.0	South	408.000000
39	2023-02-09	A	62.0	Product1	155.0	West	155.000000
40	2023-02-10	B	15.0	Product1	578.0	East	578.000000
41	2023-02-11	C	97.0	Product1	256.0	East	256.000000
42	2023-02-12	A	93.0	Product3	164.0	West	164.000000
43	2023-02-13	A	43.0	Product3	949.0	East	949.000000
44	2023-02-14	A	96.0	Product3	830.0	East	830.000000
45	2023-02-15	B	99.0	Product2	599.0	West	599.000000
46	2023-02-16	B	6.0	Product1	938.0	South	938.000000
47	2023-02-17	B	69.0	Product3	143.0	West	143.000000
48	2023-02-18	C	65.0	Product3	182.0	North	182.000000
49	2023-02-19	C	11.0	Product3	708.0	North	708.000000

[12]:
```
df.dtypes
```

[12]:
```
Date            object
Category        object
Value          float64
Product         object
```

```
Sales           float64
Region           object
Sales_fillNA    float64
dtype: object
```

[13]:
```
## Renaming Columns
df=df.rename(columns={'Sale Date':'Sales Date'})
df.head()
```

[13]:

	Date	Category	Value	Product	Sales	Region	Sales_fillNA
0	2023-01-01	A	28.0	Product1	754.0	East	754.0
1	2023-01-02	B	39.0	Product3	110.0	North	110.0
2	2023-01-03	C	32.0	Product2	398.0	East	398.0
3	2023-01-04	B	8.0	Product1	522.0	East	522.0
4	2023-01-05	B	26.0	Product3	869.0	North	869.0

[16]:
```
## change datatypes
df['Value_new']=df['Value'].fillna(df['Value'].mean()).astype(int)
df.head()
```

[16]:

	Date	Category	Value	Product	Sales	Region	Sales_fillNA	\
0	2023-01-01	A	28.0	Product1	754.0	East	754.0	
1	2023-01-02	B	39.0	Product3	110.0	North	110.0	
2	2023-01-03	C	32.0	Product2	398.0	East	398.0	
3	2023-01-04	B	8.0	Product1	522.0	East	522.0	
4	2023-01-05	B	26.0	Product3	869.0	North	869.0	

	Value_new	New Value
0	28	56.0
1	39	78.0
2	32	64.0
3	8	16.0
4	26	52.0

[17]:
```
df['New Value']=df['Value'].apply(lambda x:x*2)
df.head()
```

[17]:

	Date	Category	Value	Product	Sales	Region	Sales_fillNA	\
0	2023-01-01	A	28.0	Product1	754.0	East	754.0	
1	2023-01-02	B	39.0	Product3	110.0	North	110.0	
2	2023-01-03	C	32.0	Product2	398.0	East	398.0	
3	2023-01-04	B	8.0	Product1	522.0	East	522.0	
4	2023-01-05	B	26.0	Product3	869.0	North	869.0	

	Value_new	New Value
0	28	56.0
1	39	78.0

```
2          32       64.0
3           8       16.0
4          26       52.0
```

[18]:
```
## Data Aggregating And Grouping
df.head()
```

[18]:
```
         Date Category  Value   Product  Sales Region  Sales_fillNA  \
0  2023-01-01        A   28.0  Product1  754.0   East         754.0
1  2023-01-02        B   39.0  Product3  110.0  North         110.0
2  2023-01-03        C   32.0  Product2  398.0   East         398.0
3  2023-01-04        B    8.0  Product1  522.0   East         522.0
4  2023-01-05        B   26.0  Product3  869.0  North         869.0

   Value_new  New Value
0         28       56.0
1         39       78.0
2         32       64.0
3          8       16.0
4         26       52.0
```

[20]:
```
grouped_mean=df.groupby('Product')['Value'].mean()
print(grouped_mean)
```

```
Product
Product1    46.214286
Product2    52.800000
Product3    55.166667
Name: Value, dtype: float64
```

[21]:
```
grouped_sum=df.groupby(['Product','Region'])['Value'].sum()
print(grouped_sum)
```

```
Product   Region
Product1  East      292.0
          North       9.0
          South     100.0
          West      246.0
Product2  East       56.0
          North     127.0
          South     181.0
          West      428.0
Product3  East      202.0
          North     203.0
          South     215.0
          West      373.0
Name: Value, dtype: float64
```

[22]:
```
df.groupby(['Product','Region'])['Value'].mean()
```

[22]:
```
Product   Region
Product1  East      41.714286
          North      4.500000
          South     50.000000
          West      82.000000
Product2  East      28.000000
          North     63.500000
          South     60.333333
          West      53.500000
Product3  East      50.500000
          North     40.600000
          South     71.666667
          West      62.166667
Name: Value, dtype: float64
```

[23]:
```
## aggregate multiple functions
groudped_agg=df.groupby('Region')['Value'].agg(['mean','sum','count'])
groudped_agg
```

[23]:
```
              mean     sum  count
Region
East     42.307692   550.0     13
North    37.666667   339.0      9
South    62.000000   496.0      8
West     61.588235  1047.0     17
```

[25]:
```
### Merging and joining Dataframes
# Create sample DataFrames
df1 = pd.DataFrame({'Key': ['A', 'B', 'C'], 'Value1': [1, 2, 3]})
df2 = pd.DataFrame({'Key': ['A', 'B', 'D'], 'Value2': [4, 5, 6]})
```

[27]:
```
df1
```

[27]:
```
  Key  Value1
0   A       1
1   B       2
2   C       3
```

[28]:
```
df2
```

[28]:
```
  Key  Value2
0   A       4
1   B       5
2   D       6
```

[30]:
```
## Merge Dataframme on the 'Key Columns'
pd.merge(df1,df2,on="Key",how="inner")
```

[30]:

	Key	Value1	Value2
0	A	1	4
1	B	2	5

[31]:
```
pd.merge(df1,df2,on="Key",how="outer")
```

[31]:

	Key	Value1	Value2
0	A	1.0	4.0
1	B	2.0	5.0
2	C	3.0	NaN
3	D	NaN	6.0

[32]:
```
pd.merge(df1,df2,on="Key",how="left")
```

[32]:

	Key	Value1	Value2
0	A	1	4.0
1	B	2	5.0
2	C	3	NaN

[33]:
```
pd.merge(df1,df2,on="Key",how="right")
```

[33]:

	Key	Value1	Value2
0	A	1.0	4
1	B	2.0	5
2	D	NaN	6